



How to Successfully Achieve Application Lifecycle Management

Application lifecycle management (ALM) seems to have an identity crisis. The constant state of flux around the various tools and components that attempt to help automate ALM practices are what really drive confusion around the term, ALM.

This SearchSoftwareQuality.com expert E-guide explores the many different masks that ALM wears and how to make sure your solutions wears the right one for your organization. Read this E-Guide to help you achieve successful application lifecycle management.

Sponsored By:

ASG[®]
Software Solutions



How to Successfully Achieve Application Lifecycle Management

Table of Contents

[Understanding application lifecycle management's complex identity](#)

[Eleven steps to kickoff application lifecycle management](#)

[Resources from ASG](#)

Understanding application lifecycle management's complex identity

By Mike Jones

Application lifecycle management (ALM) seems to have an identity crisis, or, at the very least, appears to take a cue from a "Scooby Doo" villain. Once a common definition begins to gain acceptance, the technology pulls off its mask and shows itself to be something completely different. Even if you've spent your lifetime in the application development space, it's still easy to be confused by such a nebulous and (sometimes) schizophrenic term.

What really drives the confusion around the term ALM is the constant state of flux around the various tools and components that attempt to help automate ALM practices. I believe the key here is to keep the ultimate goal of ALM in focus: to improve application development and maintenance projects, making these projects more predictable and transparent to benefit the development team and the business as a whole.

Agreeing on the end goal of ALM is an important step but only marginally helps us define the term. To keep digging for a solid definition, let's examine the tools and technologies that are often classified as "ALM." According to various industry analysts and software vendors, at any given time, the following classes of tools are categorized as ALM tools including:

- Requirements analysis & management tools
- Modeling and design tools
- Project management tools
- Configuration management tools
- Build management tools
- Software testing
- Release management tools
- Issue management tools

The above list provides a good hint into the complexity of really managing the application lifecycle, especially if you're trying to define ALM and implement some kind of solution for your business. For most, buying all of the tools above is simply not reasonable. For others,

the problem lies in taking one or more tools from each category and trying to make them work together. So what technologies should you consider in helping define your ALM solution?

It's not so much about picking specific products for your ALM implementation, but rather finding the right process and framework. To get started I recommend you begin defining the key process and organizational constraints across the entire application lifecycle in your company-- don't just focus on a specific stage of the cycle.

So now with the end goal of ALM in mind and your application lifecycle processes understood, you can take the next step and begin to look at the underlying tools and technology that should support it. But just what is it?

First, let's consider the current definition of ALM -- let's look at the description offered up by TechTarget:

"ALM is a set of disciplines that together govern the process of turning business ideas into software."

My take? This definition works fine as long as we are not short sighted on how broad the definition needs to be. We cannot forget the operational and measurement aspects of the application development process. Thus here is my tweak on the definition above and what I like to call ALM 2010: "ALM 2010 is the set of disciplines, processes and tools which impact application delivery, application operations and end user teams in the effort to successfully deliver and maintain application software."

So, with this definition I would like to highlight three key needs of ALM 2010 that should not be overlooked. In addition, for any modern enterprise, it's more than just implementing a static ALM process and set of supporting tools. You need to be prepared for your ALM approach to evolve and understand what internal (not just technology) processes will be affected as your application landscape evolves faster than it has before. With this in mind, I propose your ALM 2010 approach must encompass the following:

- Seamless cross team collaboration, which includes requirements, issues, code changes and continuous integration
- A 'build to change' philosophy
- Fully integrated testing and performance management capabilities

For ALM to live up to its hype in 2010 (and potentially beyond), each of these needs must be successfully addressed. So what does each entail?

Seamless cross team collaboration: I propose that you need to invest in the processes and supporting tools to make sure you are able to drive application delivery and maintenance using a diverse set of team members. The modern shop must coordinate distributed development resources in such a way that change impact is minimized and code management is simplified. You will need to easily collect feedback from quality assurance and end user team members spread across different locations, and the feedback must be collected as close to the conception point as possible and automated to eliminate ambiguity. Operational team members must have their processes addressed and be included in the ALM definition for you to be successful. Last but not least, each of these team members needs to have transparent access to information regarding application status to avoid confusion and breakdown of the processes.

"Build to change" philosophy: A 'build to change' philosophy requires you to adopt some form of agile development process. I recommend you do this for both new development and maintenance activities. By adopting an agile approach you will be forced to look into streamlining key aspects of your development tools and how to eliminating the risk associated with rapid change. This will more than likely impact everything from your IDE to your build and configuration management tools not to mention how you capture requirements and feedback.

Fully integrated testing and performance management capabilities: I call this out specifically as testing and performance management are often overlooked in many ALM situations. If you are going to adopt a 'build to change' philosophy as part of your ALM 2010 approach these items must be seamlessly integrated into your process and platform. If not, you will miss quality objectives by not having the time to adequately test and monitor for performance issues.

ALM 2010 is no longer just about bringing a business idea through to a software application - it's so much more than that. ALM 2010 has to deal with more shareholders having input in the development process, the need to build and change rapidly and the tools to make this happen while reducing risk and increasing transparency.

Is my ALM 2010 definition off the mark? Perhaps-- but I propose you use it as a starting point to provide your specific ALM definition with the greatest flexibility. With this simple yet broad ALM 2010 definition, it can really wear whatever mask you would like. Need more collaboration? Add in the proper tools and policies to your solution. Want tighter process management? Make the appropriate tweaks. ALM wears many masks-- make sure your solution wears the right one for your organization.

Eleven steps to kickoff application lifecycle management

By John Scarpino, D. Sc.

Application lifecycle management (ALM) is not to be confused with the software development lifecycle (SDLC). While ALM is embedded within the SDLC, it is actually a more detailed interface that dictates how application development is conducted. In today's world, we can manage our application development in several different ways. Each company, manager and department has its own method of supervising what they do and how they conform with company standards when developing an application. For example, a company that develops software using a waterfall SDLC is quite different than a company that develops software using an agile SDLC.

Software tools are now coming out with ALM options, but though a tool may have an ALM component, each company will use it differently.

The methods through which a software development company uses (or doesn't use) ALM is indicative of how well its end product will meet criteria for being successful. ALM processes should be defined and agreed upon by all internal and external stakeholders involved during and after application development. A single person or role can not take the place of a multi-faceted process. Sometimes projects fall into the reins of a project manager who dictates process based on his or her view of what process should be-- though he or she is more concerned with scope, time and cost. The impact of process and ALM (how well the requirements meet the expectations of the customers) is the quality assurance department's responsibility.

Below are 11 steps that can help you achieve successful application lifecycle management.

1. **Define the roles that should be part of the ALM.** Define who is responsible for each role in information technology, software technology, quality assurance, software testing, the lines of business, and on the customer's side.

2. **Define the goals and objectives of each role.** If this step isn't followed through, then the details of each role will not be understood and the question of "what do I do?" will never be clear. This step helps to avoid the "he-said / she said" and finger-pointing.
3. **Anticipate a certain amount of internal-process dependency.** Outline and illustrate what work and which roles will depend upon your actions and deliverables. If this is not clearly documented, other people working on the project may assume that time is not important, or deliverables can be delayed or covered up if done incorrectly. A proper ALM process ensures that all levels of work are done correctly and follow quality standards.
4. **Develop a risk and contingency plan.** Every role and each step in the process has the potential to make waves if something goes wrong. Make sure you think about how a missed deadline or objective could be rectified without a major disruption to the overall ALM process. If the risks are not understood from the beginning of the project, then you will be constantly working in "ticking time bomb" environment. Problems affecting time, functionality, quality, outputs, security, performance, schedule and cost can all add to the risk of the end product, the application and the customers. Thus, it is always better to be proactive with a constructive, reusable lifecycle.
5. **Prepare a process.** It is important that you outline a flowing pathway of communication and documentation. This is where the SDLC comes into play and defines how a company and/or department will operate.
6. **Foresee each individual's potential output.** Understand the capabilities of each person on your team and set expectations accordingly.
7. **Plan ... plan again ... and plan some more!** If you couldn't already tell, detailed planning is a must for ALM to be successful. Each role must be defined so that when a project is started each role's objectives, dependencies and needs are known.
8. **Know how communication affects your clients and other external stakeholders.** When you interface with a client and that client expects something in return, how will you involve them in the process? Communication between you and your customers is crucial, since the earlier your clients are involved the more they will feel a part of the process; they will be more likely to accept your thoughts and ideas.

9. **Set up quality assurance checkpoints to ensure your ALM process is working.** This step is usually conducted by a quality analyst, whose job is to ensure that all processes are running like clockwork. (A quality analyst is not to be confused with a software tester whose role should have already been defined in step number one.) When a project is being run by a project manager, they often want to also manage ALM – but if this occurs it is difficult to provide good quality assurance since there is little or no opportunity for outside evaluation. Quality checkpoints by a quality analyst help create balance within process.
10. **Work toward continual improvement.** At the end of the first lifecycle attempt, it is QA's responsibility to delineate what went right and what went wrong in order to correct any bumps in the process. A "lessons learned" document should be prepared and presented to all of the key stakeholders, internally and externally, and discussion should ensue about how to improve the process for the next attempt. If not, then a process truly does not exist. This tends to happen if one person or team is "running the show" rather than a collective group.
11. **Provide governance.** After a constructive baseline is set, all process changes must go through a formal procedure to ensure new rules and changes within the company's adoption of the ALM process. This is why step number nine (quality assurance checkpoints) is so important. QA monitors the processes that put into place. Governance then works hand-in-hand with regulatory in-house and external audits for standards and assurance.

A successful ALM process is one that is understood by the all of the individuals involved in the project – not one of which is invisible, including external stakeholders such as customers. Every company is unique and the same can be said for their processes and technology; but if software development organizations start using the 11 steps outlined above, they will be on their way to better ALM design and a more successful future.

About the author: Dr. John Scarpino is director of quality assurance and a university instructor in Pittsburgh. You may contact him at Scarpino@RMU.edu.

Resources from ASG



[ASG Applications Management Solutions](#)

[ASG-becubic: for Application Discovery and Understanding](#)

About ASG

ASG provides a full range of practical software solutions that help IT organizations lower costs, save time, and make proactive decisions that drive business success. Best known for its broad portfolio of best-value, results-driven technologies, ASG partners with 85 percent of the world's largest companies to optimize IT service delivery in both mainframe and distributed environments. Founded in 1986, ASG is a privately held global company based in Naples, Fla., with more than 70 offices worldwide.